LAPORAN PRAKTIKUM

MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun oleh:
Bayu Kuncoro Adi
NIM: 2311102031

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2024

BAB I

TUJUAN PRAKTIKUM

A. TUJUAN PRAKTIKUM

1. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
2. Praktikan dapat membuat linked list circular dan non circular.
3. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

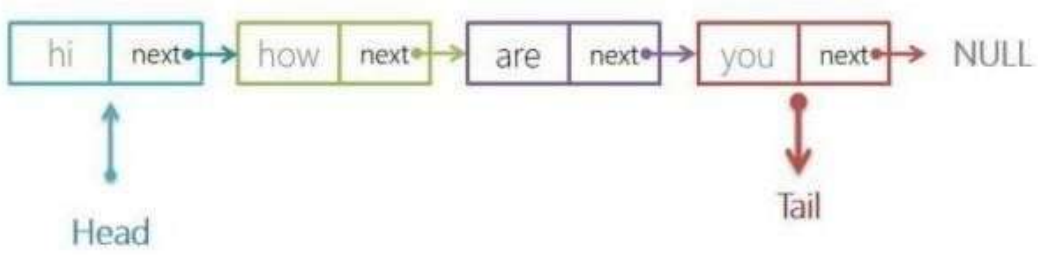
BAB II

DASAR TEORI

A. DASAR TEORI

1. Linked List Non Circular

Linked list non circular merupakan *linked list* dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada *Linked List* ini selalu bernilai '***NULL***' sebagai pertanda data terakhir dalam *list*-nya. *Linkedlist non circular* dapat digambarkan sebagai berikut.



Gambar 1 *Single Linked List Non Circular*

OPERASI PADA LINKED LIST NON CIRCULAR

1. Deklarasi Simpul (Node)

```
struct node
{
        int data; node
        *next;
};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
node *head, *tail;void
init()
{
        head = NULL;
        tail = NULL;
};
```

3. Pengecekan Kondisi Linked List

```
bool isEmpty()
{
	if (head == NULL && tail == NULL)
	{
		return true;
	}
	else
	{
		return false;
	}
}
```

4. Penambahan Simpul (Node)

```
void insertBelakang(string data User)
{
	if (isEmpty() == true)
	{
		node *baru = new node;baru-
		>data = data User;head = baru;
		tail = baru;
		baru->next = NULL;
	}

	else
	{
		node *baru = new node;baru-
		>data = data User;baru->next
		= NULL; tail->next = baru;
		tail = baru;
	}
};
```

5. Penghapusan Simpul (Node)

```
void hapusDepan()
{
        if (isEmpty() == true)
        {
                cout << "List kosong!" << endl;
        }
        else
        {
                node *helper;
                helper = head;
                if (head == tail)
                {
                        head = NULL; tail =
                        NULL; delete helper;
                }
                else
                        head = head->next;
                helper->next = NULL; delete
                helper;
           }
     }
}
```

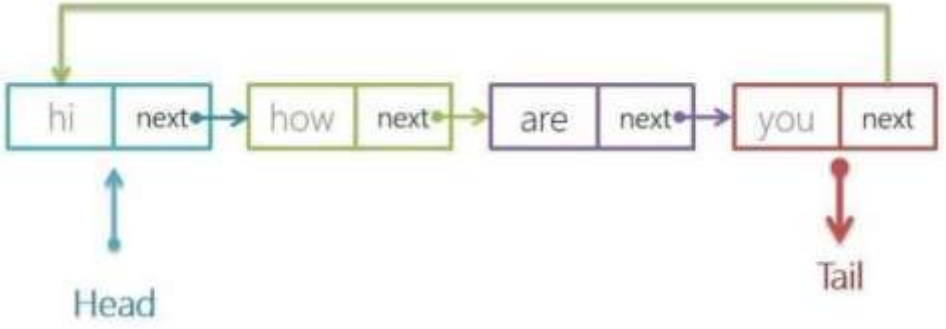
6. Tampil Data Linked List

```
void tampil()
{
	if (isEmpty() == true)
	{
		cout << "List kosong!" << endl;
	}
	else
	{
		node *helper;
		helper = head;
		while (helper != NULL)
		{
			cout << helper->data << ends;helper =
			helper->next;
		}
	}
}
```

2. Linked List Circular

Linked list circular merupakan *linked list* yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai '***NULL***', tetapi terhubung dengan node pertama (head). Saat menggunakan *linked list circular* kita membutuhkan *dummy node* atau node pengecoh yang biasanya dinamakan dengan node *current* supaya program dapat berhenti menghitung data ketika node *current* mencapai node pertama (head).

Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi. *Linked list circular* dapat digambarkan sebagai berikut.



Gambar 2 *Single Linked List Circular*

OPERASI PADA LINKED LIST CIRCULAR

1. Deklarasi Simpul (Node)

```
struct Node
{
        string data;
        Node *next;
};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
Node *head, *tail, *baru, *bantu, *hapus;

void init()
{
        head = NULL;
        tail = head;
}
```

3. Pengecekan Kondisi Linked List

```
int isEmpty()
{
        if (head == NULL) return 1;
                // true
        else
                return 0; // false
}
```

4. Pembuatan Simpul (Node)

```
void buatNode(string data)
{
        baru = new Node; baru-
        >data = data;baru->next
        = NULL;
}
```

5. Penambahan Simpul (Node)

```
// Tambah Depan
void insertDepan(string data)
{
        // Buat Node baru
        buatNode(data);

        if (isEmpty() == 1)
        {
                head = baru;tail
                = head;
                baru->next = head;
        }
        else
        {
                while (tail->next != head)
                {
                        tail = tail->next;
                }
                baru->next = head;head
                = baru;
                tail->next = head;
        }
}
```

6. Penghapusan Simpul (Node)

```
void hapusBelakang()
{
	if (isEmpty() == 0)
	{
		hapus = head;tail
		= head;
		if (hapus->next == head)
		{
			head = NULL;
			tail = NULL;

			delete hapus;
		}
		else
		{
			while (hapus->next != head)
			{
				hapus = hapus->next;
			}
			while (tail->next != hapus)
			{
				tail = tail->next;
			}
			tail->next = head;hapus-
			>next = NULL;

			delete hapus;
		}
}
```

7. Menampilkan Data Linked List

```
void tampil()
{
        if (isEmpty() == 0)
        {
                tail = head;do
                {
                        cout << tail->data << ends;tail = tail-
                        >next;
                } while (tail != head);cout <<
                endl;
        }
}
```

BAB III

GUIDED

1. Guided 1

Linked List Non Circular

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    int data;
    Node *next;
};
Node *head;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}
// Tambah Depan
void insertDepan(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
```

```
}
// Tambah Belakang
void insertBelakang(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}
// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
// Tambah Tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        // tranversing
        bantu = head;
```

```
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
        }
            nomor++;
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
```

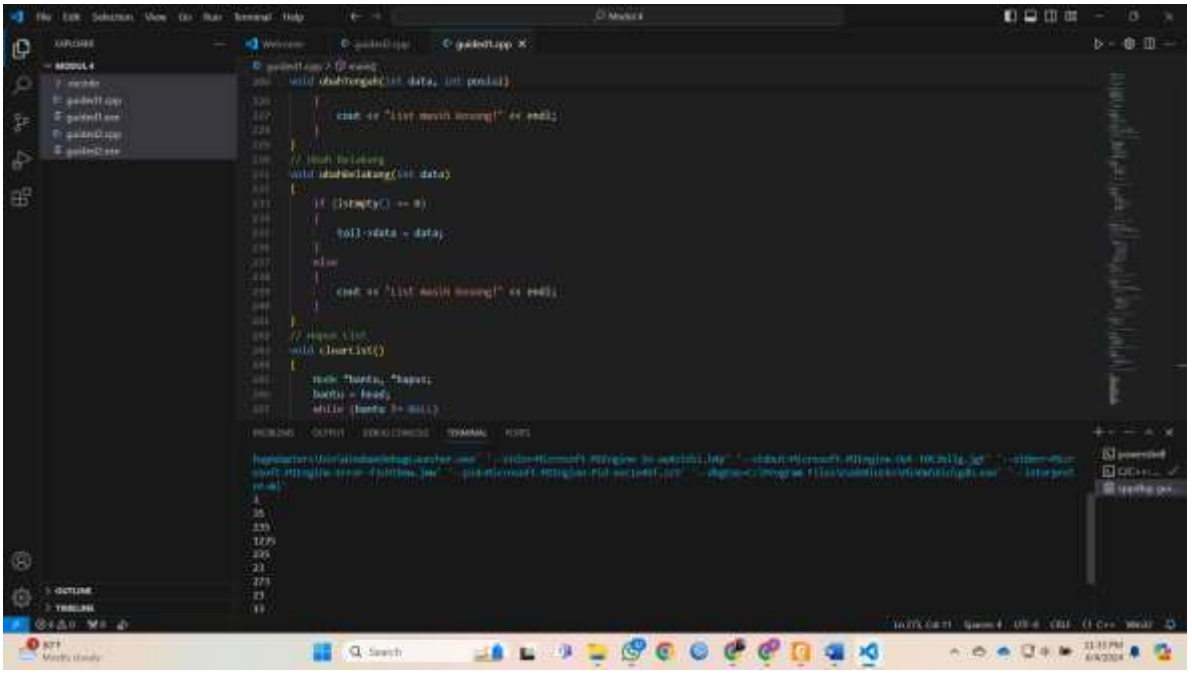
```
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                sebelum = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        sebelum->next = bantu;
        delete hapus;
    }
}
// Ubah Depan
void ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
```

```
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
        }
        else
        {
            cout << "Posisi bukan posisi tengah" << endl;
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus List
```

```
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
```

```
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}
```

Screenshoot Program:



Deskripsi Program

Program ini adalah implementasi dari struktur data linked list satu arah (non-circular) menggunakan bahasa pemrograman C++. Program ini memiliki fungsi-fungsi dasar seperti inisialisasi linked list, penambahan elemen di depan, di belakang, dan di tengah, penghapusan elemen di depan, di belakang, dan di tengah, serta pengubahan nilai elemen di depan, di belakang, dan di tengah. Selain itu, program ini juga memiliki fungsi untuk menghitung jumlah elemen dalam linked list dan untuk membersihkan seluruh isi linked list. Melalui fungsi main, program menunjukkan penggunaan fungsi-fungsi tersebut dengan contoh penambahan, penghapusan, dan pengubahan elemen dalam linked list, serta menampilkan isi linked list setelah setiap operasi dilakukan.

2. **Guided 2**

Linked List Circular

```
#include <iostream>

using namespace std;

// Deklarasi Struct Node

struct Node
{
    string data;
    Node *next;
};

Node *head, *tail, *baru, *bantu, *hapus;

// Inisialisasi node head & tail
void init()
{
    head = NULL;
    tail = head;
}

// Pengecekan isi list
int isEmpty()
{
    if (head == NULL)
    {
        return 1; // true
    }
    else
    {
        return 0; // false
    }
}

// Buat Node Baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}

// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL)
```

```
    {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

// Tambah Belakang
void insertBelakang(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        tail->next = baru;
```

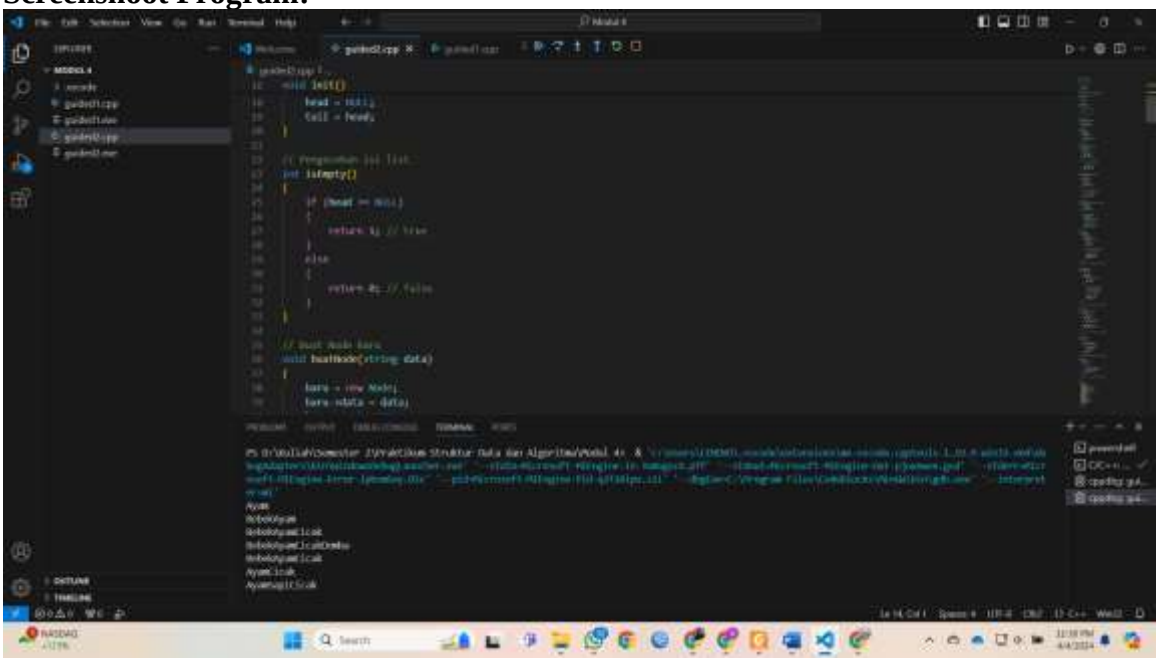
```
        baru->next = head;
    }
}

// Tambah Tengah
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Depan
void hapusDepan()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
```

```
            hapus->next = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus Belakang
void hapusBelakang()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 0)
    {
        // transversing
```

```
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList()
{
    if (head != NULL)
    {
        hapus = head->next;
        while (hapus != head)
        {
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
    else
    {
```

```
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();

    return 0;
}
```

Screenshoot Program:



Deskripsi Program:
Program ini merupakan implementasi dari struktur data linked list sirkular (circular) dengan menggunakan bahasa pemrograman C++. Setiap node dalam linked list menyimpan sebuah string dan memiliki pointer yang menunjuk ke node berikutnya. Program ini memiliki fungsi-fungsi dasar untuk melakukan operasi pada linked list, seperti penambahan elemen di depan, di belakang, dan di tengah, penghapusan elemen di depan, di belakang, dan di tengah, serta membersihkan seluruh isi linked list. Fungsi main digunakan untuk menunjukkan penggunaan fungsi-fungsi tersebut dengan contoh penambahan, penghapusan, dan pengubahan elemen dalam linked list, serta menampilkan isi linked list setelah setiap operasi dilakukan.

UNGUIDED

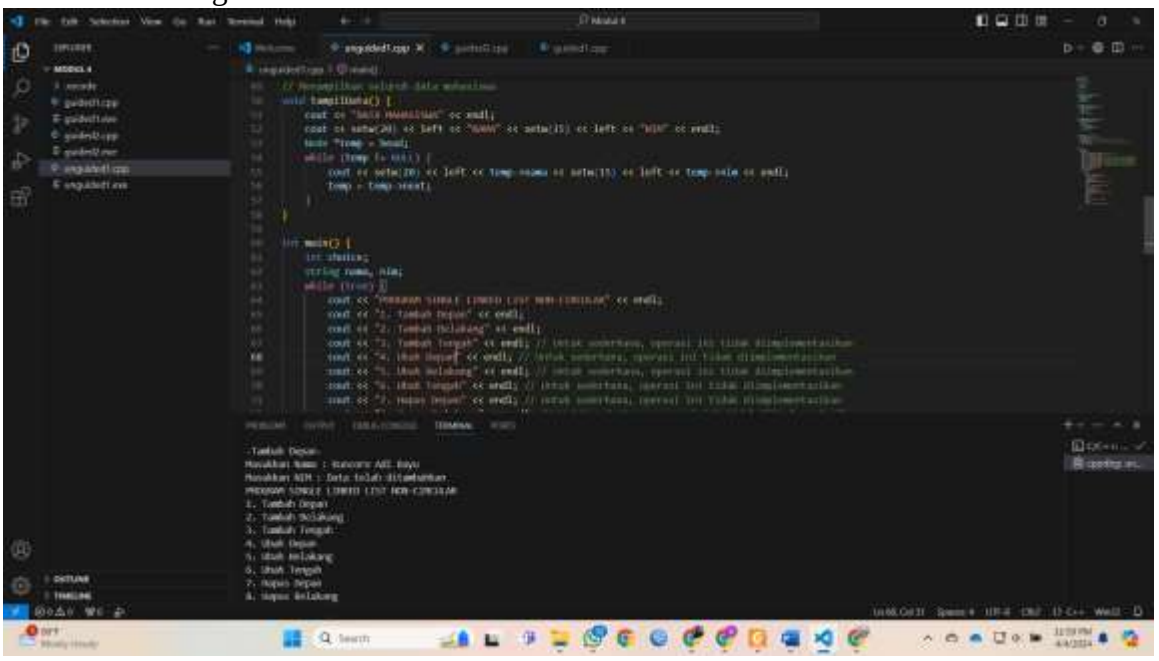
1. Unguided 1

```cpp
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

// Deklarasi Struct Node
struct Node {
    string nama;
    string nim;
    Node *next;
};

Node *head = NULL;

// Fungsi untuk membuat node baru
Node* buatNode(string nama, string nim) {
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    return baru;
}

// Menambahkan node di depan
void tambahDepan(string nama, string nim) {
    Node *baru = buatNode(nama, nim);
    if (head == NULL) {
        head = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Menambahkan node di belakang
void tambahBelakang(string nama, string nim) {
    Node *baru = buatNode(nama, nim);
    if (head == NULL) {
        head = baru;
    } else {
        Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = baru;
    }
}
```

```
// Menampilkan seluruh data mahasiswa
void tampilData() {
    cout << "DATA MAHASISWA" << endl;
    cout << setw(20) << left << "NAMA" << setw(15) << left << "NIM" <<
endl;
    Node *temp = head;
    while (temp != NULL) {
        cout << setw(20) << left << temp->nama << setw(15) << left <<
temp->nim << endl;
        temp = temp->next;
    }
}

int main() {
    int choice;
    string nama, nim;
    while (true) {
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
        cout << "1. Tambah Depan" << endl;
        cout << "2. Tambah Belakang" << endl;
        cout << "3. Tambah Tengah" << endl;
        cout << "4. Ubah Depan" << endl;
        cout << "5. Ubah Belakang" << endl;
        cout << "6. Ubah Tengah" << endl;
        cout << "7. Hapus Depan" << endl;
        cout << "8. Hapus Belakang" << endl;
        cout << "9. Hapus Tengah" << endl;
        cout << "10. Hapus List" << endl;
        cout << "11. TAMPILKAN" << endl;
        cout << "0. KELUAR" << endl;
        cout << "Pilih Operasi : ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "-Tambah Depan-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                tambahDepan(nama, nim);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 2:
                cout << "-Tambah Belakang-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
```

```
                tambahBelakang(nama, nim);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 11:
                tampilData();
                break;
            case 0:
                cout << "Terima kasih!" << endl;
                exit(0);
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    }

    return 0;
}
```

Screenshoot Program:



Deskripsi Program:

Program ini merupakan implementasi dari linked list non-circular dalam bahasa pemrograman C++. Program ini menyediakan menu interaktif untuk pengguna yang memungkinkan mereka untuk menambahkan data mahasiswa di depan atau di belakang linked list serta melihat semua data mahasiswa yang telah dimasukkan. Program ini memanfaatkan struktur data linked list untuk menyimpan nama dan NIM mahasiswa dalam setiap node. Setiap kali pengguna memilih operasi tambah, data mahasiswa baru akan dimasukkan ke dalam linked list sesuai dengan pilihan mereka. Kemudian, saat pengguna memilih untuk melihat data, program akan menampilkan semua nama dan NIM mahasiswa yang telah dimasukkan sebelumnya. Program ini sederhana dan masih memiliki ruang untuk penambahan fungsionalitas tambahan seperti pengubahan atau penghapusan data, sesuai dengan kebutuhan pengguna.

2. Unguided 2

```cpp
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

// Deklarasi Struct Node
struct Node {
    string nama;
    string nim;
    Node *next;
};

Node *head = NULL;

// Fungsi untuk membuat node baru
Node* buatNode(string nama, string nim) {
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    return baru;
}

// Menambahkan node di depan
void tambahDepan(string nama, string nim) {
    Node *baru = buatNode(nama, nim);
    if (head == NULL) {
        head = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Menambahkan node di belakang
void tambahBelakang(string nama, string nim) {
    Node *baru = buatNode(nama, nim);
    if (head == NULL) {
        head = baru;
    } else {
        Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = baru;
    }
}
```

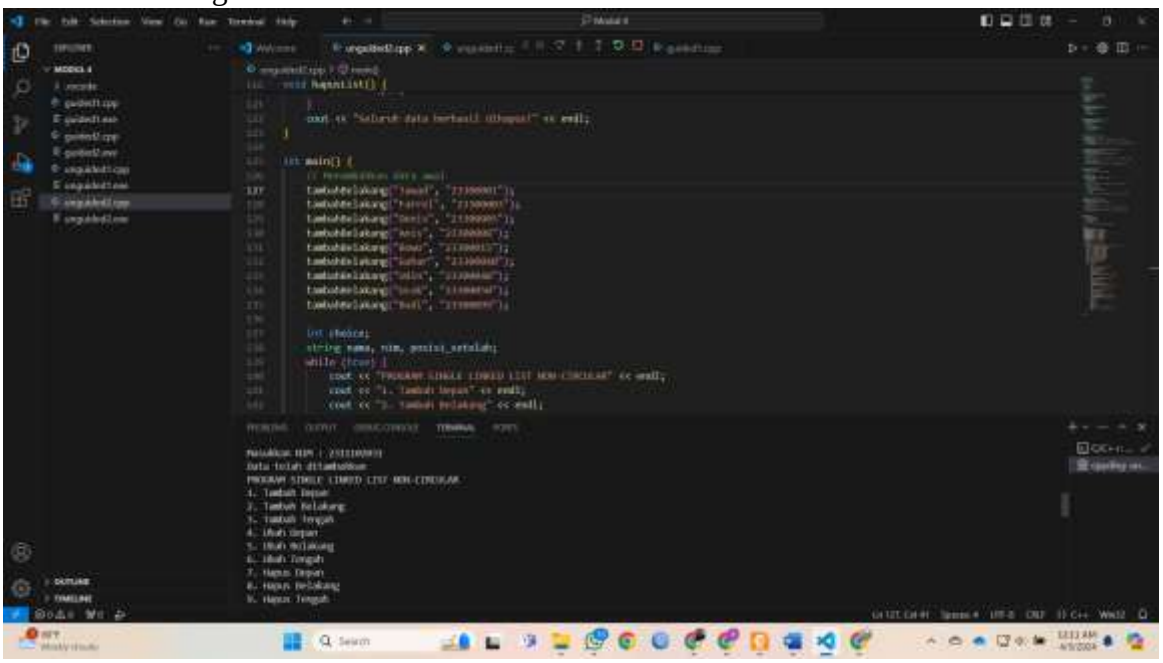
```
// Menampilkan seluruh data mahasiswa
void tampilData() {
    cout << "DATA MAHASISWA" << endl;
    cout << setw(20) << left << "NAMA" << setw(15) << left << "NIM" <<
endl;
    Node *temp = head;
    while (temp != NULL) {
        cout << setw(20) << left << temp->nama << setw(15) << left <<
temp->nim << endl;
        temp = temp->next;
    }
}

// Tambahkan node di tengah setelah node tertentu
void tambahTengah(string nama, string nim, string posisi_setelah) {
    Node *baru = buatNode(nama, nim);
    Node *temp = head;
    while (temp != NULL && temp->nama != posisi_setelah) {
        temp = temp->next;
    }
    if (temp == NULL) {
        cout << "Node dengan nama " << posisi_setelah << " tidak
ditemukan!" << endl;
    } else {
        baru->next = temp->next;
        temp->next = baru;
        cout << "Data telah ditambahkan setelah " << posisi_setelah <<
endl;
    }
}

// Hapus node dengan nama tertentu
void hapusNode(string nama) {
    if (head == NULL) {
        cout << "Linked list kosong!" << endl;
        return;
    }
    Node *temp = head;
    Node *sebelum = NULL;
    while (temp != NULL && temp->nama != nama) {
        sebelum = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        cout << "Node dengan nama " << nama << " tidak ditemukan!" <<
endl;
    } else {
        if (sebelum == NULL) {
```

```
            head = temp->next;
        } else {
            sebelum->next = temp->next;
        }
        delete temp;
        cout << "Data dengan nama " << nama << " berhasil dihapus!" <<
endl;
    }
}

// Ubah data node dengan nama tertentu
void ubahData(string nama, string nim) {
    Node *temp = head;
    while (temp != NULL && temp->nama != nama) {
        temp = temp->next;
    }
    if (temp == NULL) {
        cout << "Node dengan nama " << nama << " tidak ditemukan!" <<
endl;
    } else {
        temp->nim = nim;
        cout << "Data dengan nama " << nama << " berhasil diubah!" <<
endl;
    }
}

// Hapus seluruh data
void hapusList() {
    while (head != NULL) {
        Node *hapus = head;
        head = head->next;
        delete hapus;
    }
    cout << "Seluruh data berhasil dihapus!" << endl;
}

int main() {
    // Menambahkan data awal
    tambahBelakang("Jawad", "23300001");
    tambahBelakang("Farrel", "23300003");
    tambahBelakang("Denis", "23300005");
    tambahBelakang("Anis", "23300008");
    tambahBelakang("Bowo", "23300015");
    tambahBelakang("Gahar", "23300040");
    tambahBelakang("Udin", "23300048");
    tambahBelakang("Ucok", "23300050");
    tambahBelakang("Budi", "23300099");

    int choice;
    string nama, nim, posisi_setelah;
    while (true) {
```

```
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
        cout << "1. Tambah Depan" << endl;
        cout << "2. Tambah Belakang" << endl;
        cout << "3. Tambah Tengah" << endl;
        cout << "4. Ubah Depan" << endl;
        cout << "5. Ubah Belakang" << endl;
        cout << "6. Ubah Tengah" << endl;
        cout << "7. Hapus Depan" << endl;
        cout << "8. Hapus Belakang" << endl;
        cout << "9. Hapus Tengah" << endl;
        cout << "10. Hapus List" << endl;
        cout << "11. TAMPILKAN" << endl;
        cout << "0. KELUAR" << endl;
        cout << "Pilih Operasi : ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "-Tambah Depan-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                tambahDepan(nama, nim);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 2:
                cout << "-Tambah Belakang-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                tambahBelakang(nama, nim);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 3:
                cout << "-Tambah Tengah-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                cout << "Masukkan Nama Posisi Setelah : ";
                cin >> posisi_setelah;
                tambahTengah(nama, nim, posisi_setelah);
                break;
            case 11:
                tampilData();
                break;
            case 0:
                hapusList();
                cout << "Terima kasih!" << endl;
```

```
                exit(0);
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    }

    return 0;
}
```

Screenshoot Program:



Deskripsi Program:

Program yang disediakan adalah sebuah aplikasi konsol yang mengimplementasikan linked list non-circular untuk menyimpan data nama dan NIM mahasiswa. Program ini memungkinkan pengguna untuk melakukan berbagai operasi seperti menambahkan data di depan atau belakang, menambahkan data di tengah setelah node tertentu, mengubah data, menghapus data, serta menampilkan seluruh data yang telah dimasukkan. Setiap operasi diimplementasikan melalui fungsi-fungsi yang sesuai, seperti tambahDepan(), tambahBelakang(), tambahTengah(), hapusNode(), ubahData(), dan lainnya. Data awal telah dimasukkan ke dalam linked list, kemudian pengguna dapat melakukan operasi sesuai dengan pilihan menu yang tersedia.

Program ini menggunakan struktur data linked list untuk mengelola data mahasiswa. Setiap node dalam linked list memiliki dua atribut, yaitu nama dan NIM, yang merupakan informasi dari setiap mahasiswa. Fungsi-fungsi yang disediakan memungkinkan pengguna untuk memanipulasi linked list dengan berbagai cara, seperti menambahkan data di berbagai posisi, mengubah data, menghapus data, dan menampilkan seluruh data. Implementasi ini memungkinkan pengguna untuk mengelola data mahasiswa dengan fleksibilitas sesuai dengan kebutuhan mereka. Selain itu, program ini juga memberikan pesan balasan yang informatif setiap kali operasi dilakukan untuk memberikan umpan balik kepada pengguna tentang keberhasilan atau kegagalan operasi yang dilakukan.

BAB IV

KESIMPULAN

Program-program yang disediakan merupakan implementasi dari struktur data linked list dalam bahasa pemrograman C++. Program-program tersebut mengilustrasikan penggunaan linked list non-circular dan circular untuk menyimpan dan mengelola data. Pada linked list non-circular, terdapat fungsi-fungsi dasar seperti penambahan dan penghapusan elemen di depan, di belakang, dan di tengah, serta pengubahan nilai elemen. Program ini memanfaatkan konsep linked list non-circular untuk menyimpan data mahasiswa dan menyediakan menu interaktif bagi pengguna untuk melakukan operasi-operasi tersebut.

Sementara itu, program-program yang mengimplementasikan linked list circular juga menyediakan fungsi-fungsi serupa untuk menambahkan dan menghapus elemen dalam linked list. Namun, pada linked list circular, terdapat perbedaan dalam cara penanganan elemen terakhir dan pengulangan melalui seluruh elemen dalam linked list. Program ini juga memberikan opsi kepada pengguna untuk menambahkan data di depan atau di belakang, serta menampilkan seluruh data yang telah dimasukkan.

Program-program tersebut memanfaatkan konsep linked list untuk mengelola data secara efisien. Setiap program memberikan opsi kepada pengguna untuk melakukan operasi-operasi dasar pada linked list, seperti penambahan, penghapusan, pengubahan, dan penampilan data. Pesan-pesan balasan yang dihasilkan oleh program juga memberikan informasi yang jelas tentang keberhasilan atau kegagalan operasi yang dilakukan, sehingga pengguna dapat memahami status operasi yang telah mereka lakukan dengan mudah.

Secara keseluruhan, program-program tersebut menyediakan implementasi yang baik dari konsep linked list dalam bahasa pemrograman C++. Mereka memberikan kesempatan kepada pengguna untuk memahami dan berinteraksi dengan struktur data linked list, serta memanfaatkannya untuk mengelola dan mengorganisir data dengan efisien. Dengan menggunakan program-program tersebut, pengguna dapat belajar tentang konsep linked list dan bagaimana mengimplementasikannya dalam pengembangan perangkat lunak.

DAFTAR PUSTAKA

Modul 3 Single and Double Linked List Praktikum Struktur Data dan Algoritma

Modul 6 Single dan Double Linked List diakses dari https://elektro.um.ac.id/wp-content/uploads/2016/04/ASD-Modul-6-Linked-List.pdf

Single dan Double Linked List diakses dari https://www.academia.edu/82277141/SINGLE_LINKED_LIST_DOUBLE_ENDED_LIST_DOUBLY_LINKED_LIST_CIRCULAR_LINKED_LIST_DAN_ITERATOR?f_ri=1292408

Struktur Data diakses dari https://github.com/yunusfebriansyah/struktur-data/tree/main/Circular%20Single%20Linked%20List